Assignment Objective: Build data structure experience by building a circular list ADT.

Requirements:

* The class properties:
  + Semantically, to the user, this class will behave the same as stringList from Program 1. It has all the same member functions as stringList.
  + It has two new public member functions, deleteFirst() and deleteLast; see below.
  + It has two new private member functions: incVal() and decVal(); see below.
  + However, this will be implemented in a manner such that insert() will be O(1) as well as add() is O(1).
    - To do this, two new integer member variables will be defined: first and last.
      * first will be the index of the first item on the list.
      * last will be the index of the last item on the list.
    - When the list is not full, when something is inserted, first will be decremented and the value will be put at that index. Care must be taken that if first becomes negative, it should be set to the value of capacity-1. I.e., first will wrap around to the end of the array.
    - Similarly, when an etnry is added to the list, last will be incremented by 1 and the entry will go there. If last is incremented to capacity, it will be set to 0 and the entry will be inserted there.
    - Note, insertAt() and deleteAt() become more complicated. And performance is still O(n).
* Into two files, p3.h and p3.cpp, create a class called cStringList with the following members
  + Private members:
    - string \*a // A pointer to the array into which the list values will be stored.
    - int listCapacity; // memorialize the capacity of the list
    - int listCount; An integer to record the number of entries in the list
    - int first; // to track the first entry of the list
    - int last; // to track the last entry of the list
* void incVal(int &value) // increments the passed value, wrapping its value to 0 if necessary
* void decVal(int &value) // decrements the passed value, wrapping its value to capacity-1 if necessary
  + Public members:
    - constructor cStringList(int capacity = 100) that causes the array to be sized at “capacity” entries, with a default listCapacity of 100, setting listCount to 0 and first and last also to 0.
    - destructor ~cStringList() that deletes the dynamically allocated array “a”
    - bool insert(string text) – inserts text at the beginning of the list; all other entries shift right. Returns true if text was inserted; otherwise, it returns false.
    - bool add(string text) – inserts text at the end of the list. Returns true if the text was inserted; otherwise, it returns false.
    - bool insertAt(int index, string text) – inserts text at the index “index”; values at that position need to shift right. Returns true if text is inserted. Returns false if the list was full or if the index was greater than” listCount”.
    - bool deleteAt(int index, string &text) – deletes entry at the given index; shifts the entries right of that index to the left. If the index was within the range of the list, it sets “text” to the value of the deleted entry and returns true. Otherwise, it does not change “text” and returns false.
    - Bool readAt(int index, string &text) Same as deleteAt(), but it does not delete the entry.
    - bool deleteFirst(string &text) – Behaves the same as deleteAt(0, text). But it must be independently implemented.
    - bool deleteLast(string &text) – Behaves the same as deleteAt(listCount-1, text). But it must be independently implemented.
    - void clear() – causes the list to be emptied
    - int count() const – returns the number of entries in the list
    - int getIndex(string text) const – returns the first position at which the text was found; otherwise returns -1.
    - void printIt() const – causes the list to be printed, one value per line; for each line, print the index and the value at that index.
* Compile your program: g++ p4.cpp p4m.cpp -o p4
* First submission: The highlighted portions shall be submitted as described below. Note, to do any testing, there should be a “stub” for each of the functions that are not yet completed.
* Final submission: The totality of the assignment shall be submitted as described below.
* Demonstrate that the cStringList data structure works:
  + Run your program as follows:

P3 p3input.txt > p3output.txt

* Compare your output files to the p3CorrectOutput.txt.
* **Submission process:**
  + **In class submission:**
    - Print a combined listing of only these files, in the following order: p3.h, p3.cpp, and p3Output.txt
    - Bring the listing to class when due.
  + **D2L submission:**
    - Submit these files, in a zip file, to D2L: p3.h, p3.cpp, p3Output.txt